```
Seite 1   Frei 16098   >COLBK_               <
 ;
 ;  Demo
 ;
                OR
 ;
                OR
 ;
Rtclock:        EC
 ;
Start:          LI
                AS
                TA
                CI
                ST
                AI
                ST
                TX
                SEC
                SBC Rtclock
                STA Colbk
                LDA Consol
                CMP #6 ; Start gedrueckt ?
                BNE Start ; Nein, zum Prog»
                RTS  ; Ja, Exit...
```

MacroAssembler
Version 4.3
ATARI 8 Bit
mit mindestens
128 kB RAM !

Dies ist ein
SHAREWARE-
Programm !

Macro-Assembler ATARI XE, Version 4.3.

Für alle ATARI 8-Bitter mit mindestens 128 K-Byte RAM

Hinweise:

Dieser Macro-Assembler benötigt mindestens 128 K-Byte RAM (64 KB Rechnerspeicher sowie eine RAMDISK)! Er wurde von mir entwickelt, um auch größere Quelltexte zu verarbeiten. Andere Assembler können z.B. "konfiguriert" werden (ATMAS) um mehr Quelltext (ca. 20 KB) bearbeiten zu können, haben dann aber keinen Speicher mehr für den Objectcode !

Der Quelltext kann maximal 48 K-Byte "verbraten", und zum Ausprobieren des Objectcodes bleibt immer noch genügend Platz (Der Bereich von $3000 bis $BBFF kann für Objektcode und Daten genutzt werden). Der Editor wandelt bereits bei der Eingabe den Quelltext in ein sog. Tokenformat um. Befehle wie z.B "ORG" oder "LDA" brauchen nicht mehr beim Assemblieren in einer Tabelle gesucht werden, sondern werden sofort an ihrer Codenummer erkannt!

Zur Geschwindigkeit des Assemblers: Ich habe alles Machbare versucht, um den Assembler so schnell wie möglich zu machen (logisch)! Aus Gründen, welche ich noch erläutern werde, ist der Assembler nicht so schnell wie er eigentlich sein könnte. Der Assembler benötigt zum übersetzen eines 48 KB-Quelltextes (ausprobiert) ca. 53 Sekunden (über 700 Symbole, mehr als 10 KByte Objectcode)! Bei kürzeren Programmen sinkt sogar die relative Assemblierungszeit (Wenn 48 KB in 48 Sekunden Assembliert werden, dauert es bei 16 KB nicht unbedingt auch 16 Sekunden). Der Assemblierungsvorgang wird immer in zwei Durchgängen "erledigt" (2-Pass-Assembler).

Der Speicher einer 64 KB RAMDisk wird in vier Bereiche von jeweils 16 KB aufgeteilt. Diese 16 KByte-Bereiche werden durch Bankswitching in den Speicher von $4000 bis $7FFF eingeblendet. Da jeweils nur eine 16 K-Bank eingeblendet werden kann (entweder der Editor/Assembler oder einer von drei 16 KByte-Quelltexten) ist es nicht möglich, z.B. vom Assembler mit einem "LDA (TextPt),Y" auf den Text zuzugreifen (wie es eigentlich geplant war). Das Lesen aus dem (und Schreiben zum) Quelltext muß daher durch ein Unterprogramm erledigt werden, was relativ viel Zeit kostet.

Der Macro-Assembler müßte eigentlich mit jeder Speichererweiterung zusammenarbeiten (130 XE, AtariMagazin-RAMDisk). Es gibt auch noch die CompyShop-RAMDisk, auf welcher der Assembler nicht getestet wurde (Ich kenne z. Zt. keinen, der diese besitzt).

Achtung: Das DUP.SYS-Menü muß bei einer 64 KB-RAMDisk (130 XE) im Bedarfsfall von Diskette nachgeladen werden. Der RAMDisk-Zugriff mit DOS-Befehlen wird unterbunden. 256 KB-User können die RAMDisk mitbenutzen !!!

Der Computer lädt beim BOOTen die Datei "RAMDISK.COM" in den Speicher. Ich habe dieses Programm so geändert, daß die Fast-Serial-IO-Routine einer SPEEDY-Erweiterung eingeladen wird, welche (nur!) während des BOOT-Vorganges aktiviert ist! Sollte der Computer beim Laden abstürzen, müssen Sie ein altes "RAMDISK.COM" von einer anderen Diskette Kopieren. Sie können auch, während RAMDISK.COM geladen wird, die SELECT-Taste gedrückt halten. Dies dürfte aber nicht nötig sein, da eine SPEEDY-Erweiterung erkannt wird.

Machen Sie bitte auf jeden Fall von dieser Diskette eine Sicherheitskopie !

Speicherbelegung:

Eigene Programme können den Speicher von $3000 bis $BBFF nutzen, in der ZeroPage von $A0 bis $FF !

Der Assembler belegt folgende Speicherbereiche (alle Angaben Hexadezimal) :

0080 - 009F .... für Zeiger und Zwischenspeicherung
1FF0 - 2BFF .... für System-Unterprogramme, Adresstabellen usw.
2C00 - 2FFF .... als Arbeitsbereich (Puffer, Zwischenspeicher usw.)

4000 - 4FFF (RAMDisk) Editor
5000 - 5FFF (RAMDisk) Macro-Assembler
6000 - 6FFF (RAMDisk) Verwaltung der Menü-Leiste (FileSelect, Load, Save usw.)
7000 - 7FFF (RAMDisk) Monitor

C000 - CBFF unter ROM weitere Unterprogramme, nachträglich Programmierte Extras
CC00 - CFFF unter ROM Internationaler Zeichensatz (wahlweise benutzbar)
D800 - FEFF unter ROM Symboltabelle / Resident-List (Resident-List bis dfff)
F000 - F3FF unter ROM Puffer für das Inhaltsverzeichnis der File-Select-Box
F400 - FFF0 unter ROM ToolBox-Utilities, kann durch anwachsende Symboltabelle zerstört werden, wird dann nachgeladen !

Achtung:

Es sollte vermieden werden, Interruptroutienen in den Bereich von $4000 bis $7fff zu legen, da bei einem Programmabbruch (z.B. BRK oder BREAK/SHIFT) die Interrupts nicht ausgeschaltet werden, dort jedoch die RAM-Disk eingeschaltet wird weil dort u.A. der Monitor liegt. Dann ist ein Zugriff auf den "normalen" Speicher nicht möglich und der Interruptvektor zeigt auf alles Andere als auf das eigene Programm!

Bei einigen ATARI XL's mit selbsgebauter 256 kB-Erweiterung kam es beim Test dieses Assemblers zu Fehlfunktionen und Programmabstürzen. Dies lag aber nicht an diesem Programm, sondern an Timing-Fehlern in den RAM-Bausteinen der Speichererweiterung! So kann es beim benutzen dieses Programmes passieren, da irgendwelche Störungen durch Timing-Probleme auftreten, die beim normalen Benutzen nicht auftreten oder nicht bemerkt werden (z.B. RAM-Disk Betrieb, Puffer bei Kopierprogrammen o.Ä.). Timing-Probleme können bei diesem Assembler auftreten, weil sehr häufig zwischen der Speichererweiterung und dem "normalen" Speicher hin- und hergeschaltet wird. Bei störungen ist daher erstmal (durch einen Elektroniker) das Timingverhalten der RAM's zu überprüfen, bei mir (und einigen Freunden) läuft das Programm nämlich störungsfrei.

Programmautor:

Torsten Karwoth
Hermann-von-Vechelde-Straße 19

3300 Braunschweig

Tel.:(0531) - 69 66 89

Die Version 4.3 enthält keine mir bekannten Fehler mehr. Da ich den Assembler selber regelmäßig benutzte, sind mir (fast?) alle Fehlfunktionen aufgefallen. Die erste Version (V1.0) wurde irgendwann im Herbst 1989 angefangen (Oh mein Gott, was für Fehler...)! So nach und nach wurden dann die Fehler beseitigt und die Funktionen ergänzt (Full-Screen-Editor, Menü-Leiste, Macro-Fähigkeit, Geschwindigkeitsoptimierung und ... und ... und ... !

Der Editor des Macro-Assemblers:

Nach dem laden gelangen Sie in den Editor des Macro-Assemblers. Um sich mit den Besonderheiten des Editors vertraut zu machen, geben Sie einfach 'mal folgendes Programm ein. Tippen Sie es so ein, wie es hier steht:

```
;
; Demo
;
org $3000
;
opt n,r,i
;
rtclock=$14
;
start lda vcount
asl
tax
clc
sta wsync
adc rtclock
sta colpf2
txa
sec
sbc rtclock
sta colbk
lda consol
cmp #6 Start gedrueckt ?
bne start Nein, zum Programmanfang
rts Ja, Exit...
```

Wie Sie bemerken, werden alle eingegebenen Zeilen in ihr "richtiges Format" gebracht, Sie brauchen also nicht auf Großschrift zu achten. Dem Editor ist es egal, ob Sie "start lda (ad),y" eingeben oder "START: LDA (Ad),Y" ! Achtung: Wenn Sie ein Befehlswort als Symbol verwenden wollen, müssen Sie den Doppelpunkt mit angeben ("Lda:" oder "Clc:") ! Alle eingegebenen Zeilen werden auf Fehler überprüft. Sollte die eingegebene Zeile fehlerhaft sein, wird ein Piepton sowie der Fehler im Klartext ausgegeben (Adressierungsart, Symbol falsch usw.)!

Wenn Sie das Programm vollständig eingegeben haben, Assemblieren Sie es mit CONTROL/Y. Während der Assemblierung erscheint im rechten Teil der Statuszeile "Assembling: Pass 1" bzw. "Pass 2". Allerdings hat es bei diesem kurzen Programm keinen Sinn, mit einer Uhr die Zeit messen zu wollen! Sie sehen die Angabe der Endadresse, Anzahl der Symbole sowie den freien Speicher in der Symboltabelle. "OPT N" im Listing gibt zusätzlich noch alle Symbole bzw. Macros aus. Drücken Sie eine Taste, um in den Editor zurückzukehren. Sollte ein Fehler aufgetreten sein, haben Sie vermutlich irgendwas falsch abgetippt. Um das Programm zu starten, gehen Sie mit CONTROL/P in den Monitor. geben Sie "G START" (oder "GSTART") ein und drücken Sie RETURN. Sie sehen eine sog. "Rainbow-Color-Demo", welche die Farben in jeder Rasterzeile ändert. Diese Routine "frißt" allerdings die komplette Rechenzeit des XL, da die Hardware (bzw. die CPU) mit dem Befehl "STA Wsync" zwecks synchronisation mit der Rasterzeile gestoppt wird! Um solch einen Effekt im Interrupt zu programmieren (Rasterzeilen- oder Display-List-Interrupt), lesen Sie bitte Fachliteratur zur Programmierung des ATARI! Brechen Sie das Programm ab (Mit der START-Taste oder mit RESET). Mit "X" springen Sie vom Monitor zurück in den Editor.

Hinweis:

Die Adressierungsart "Accu" wird bei einigen 6502-Assemblern auch mit einem "A" gekennzeichnet, also z.B. "ASL A". Bei diesem Macro-Assembler braucht kein "A" angegeben zu werden, es reicht "ASL". Wenn Sie "ASL A" statt "ASL" schreiben, wird das "A" als Symbol erkannt! Sollte es das Symbol "A:" geben, wird der ASL-Befehl als "ASL Absolut" in den Objektcode eingefügt! Der Editor kann von sich aus nicht wissen, ob das "A" ein Symbol ist oder für "Accu" steht ("ROL" statt "ROL A" usw.) !

Bedienung des Editors:

Der Cursor wird mit den Cursortasten gesteuert (mit CONTROL). Wenn er an den oberen oder unteren Bildschirmrand stößt, wird der Quelltext gescrollt. Mit der Tastenkombination SHIFT-CursorUp/Down kann um jeweils 16 Zeilen gescrollt werden. Der Quelltext kann wie in einem Full-Screen-Editor editiert werden.

Sollte eine Zeile versehentlich geändert worden sein oder eine geänderte Zeile fehlerhaft sein, kann die ursprüngliche Zeile mit ESCape zurückgeholt werden.

Einzelne Textzeilen werden mit SHIFT/DELETE gelöscht. Mehrere Zeilen löschen Sie besser mit der Block-Kill Funktion.

Um Programmzeilen in den Quelltext einzufügen, drücken Sie SHIFT/INSERT. Sie befinden sich jetzt im Insert-Modus und an der Cursorposition wird eine Leerzeile ausgegeben.

Mit "SHIFT/CONTROL/-" kommt man zum Textanfang und mit "SHIFT/CONTROL/=" zum Textende.

Es gibt folgende CONTROL-Befehle:

Y	Assembler starten
P	Sprung in den Monitor
[	Block-Start setzen
]	Block-Ende setzen
?	Zeige Blockstart (sprung zum Blockanfang)
CLEAR	Blockmarkierung löschen
INSERT	Füge Zeichen in Zeile ein
DELETE	löscht Zeichen in Zeile
TAB	Tabulator setzten
ESC	Menü-Auswahl > siehe File "Menu.Doc"
SHIFT/-	Cursor zum Textanfang
SHIFT/=	Cursor zum Textende

Quelltext-Ebenen:

Der Quelltext wird in drei Bereiche von jeweils 16 K-Byte aufgeteilt. Diese Texte werden nacheinander Assembliert. Zwischen den 3 Quelltexten wird umgeschaltet mit:

SHIFT/CONTROL/1 .... aktiviert Seite 1
SHIFT/CONTROL/2 .... aktiviert Seite 2
SHIFT/CONTROL/3 .... aktiviert Seite 3

In der Statuszeile wird die gerade aktuelle Seite angezeigt, der Speicherplatz in dieser Seite und eine eventuelle Fehlermeldung.

Funktionen wie laden, speichern oder Block-Löschen werden über die Menüzeile angewählt.

Der Assembler:

Der Assembler wird vom Editor mit CONTROL/Y gestartet und kehrt nach der übersetzung des Quelltextes wieder zum Editor zurück. Sollte ein Fehler aufgetreten sein, so wird der Fehler im Klartext angezeigt. Nach der Assemblierung werden die Endadresse, die Anzahl der Symbole sowie der freie Speicherplatz in der Symboltabelle angezeigt.

Macro-Fähigkeit:

Es können maximal 7 numerische Parameter an Macros übergeben werden. Macros können nicht verschachtelt werden.

Macros müssen vor ihrem ersten Aufruf definiert werden! Soll ein Macro benutzt werden (bzw. aufgerufen), So müssen Sie das dem Editor "sagen", indem Sie dem Macro-Namen einen Punkt voranstellen. Wenn das Macro also "Test" heißt, so müssen Sie ".Test" eingeben. Sollte ein Symbol vor dem Macro-Aufruf stehen (z. B. "Loop_1: .Test"), so erkennt der Editor den Macro-Namen selbstständig (Sie brauchen also nur "loop_1 test" einzugeben). Wenn Sie ein Befehlswort als Macro-Namen benutzen, müssen Sie allerdings immer den Punkt mit angeben!

Definieren von Macros:

```
Name:          MAC (Anzahl der Parameter)
... Assembler-Quelltext ...
               END  ; Ende der Macro-definition
```

Beispiele für Macros:

```
Poke:          MAC 2
               LDA #?2 ; Lade Accu mit 2. Parameter (Wert)
               STA ?1 ; Schreibe Accu zum 1. Parameter (Adresse)
               END  ; Ende der Macro-Definition

DPoke          MAC 2
               LDA #<?2 ; Lade Accu mit Low-Byte vom 2. Parameter
               STA ?1 ; Schreibe Accu zur Adresse (1. Parameter)
               LDA #>?2 ; Lade Accu mit High-Byte vom 2. Parameter
               STA ?1+1 ; ...
               END

Schleife       MAC ; ohne Parameter (oder MAC 0)
               LDY #0
MacLoop@:      DEY
               BNE MacLoop@ ; @ ist der Klammeraffe für lokale Symbole!
               END
```

Benutzung von Macros im Quelltext:

```
Start:         .Poke Colpf2,$b8
               .Dpoke 560,Dlist
               .Schleife ; Macroaufruf ohne Parameter
```

".Poke Colpf2,$b8" entspricht also "LDA #$b8 / STA Colpf2". Mit Macroanweisungen können Sie z.B. Speicher sparen (Im Quelltext). Auf der Systemdiskette befindet sich eine kleine Demo (IOLIB.SRC). Sie zeigt die Benutzung von Macros und soll nur als Beispiel dienen. Allerdings müssen Macros nicht immer "Wirtschaftlich" arbeiten. So z.B. wäre es sinnvoller, das OPEN- und INPUT-Macro als Unterroutine zu Programmieren, welche (nach Parameterübergabe) mit JSR angesprungen werden kann! Jedes Benutzen des INPUT-Macros fügt den entsprechenden (langen!) Code in den Objektcode ein !!!

Rechenformel/Ausdruck:

Alle Rechenfunktionen werden in Wortbreite (16 Bit) vorgenommen. Ein Ausdruck darf aus maximal 7 Klammerebenen zur Klärung der Priorität bestehen. Folgende Operationen sind möglich (in Reihenfolge ihrer Priorität):

"#" ... Exclusiv-Oder-Verknüpfung

"&" ... logische Und-Verknüpfung
"!" ... logische Oder-Verknüpfung

"*" ... Multiplikation
"/" ... Division

"+" ... Addition
"-" ... Subtraktion

">" und "<" dienen zum selektieren des High- bzw. Low-Bytes eines Wertes und können NICHT bei geklammerten Ausdrücken benutzt werden.

Beispiel:

"Test: LDA #>Tabelle+>Offset" statt "Test: LDA #>(Tabelle+Offset)" !

Konstanten:

'A ... ASCII-Code von "A"
10 ... dezimale Konstante
$f ... hexadezimale Konstante
%1 ... binäre Konstante

Negative Konstanten:

-1, -Test ...

Assembler-Directiven:

- ORG Al(,Ap)

Festlegen der Anfangsadresse eines Programmes.

Al = logische Adresse
Ap = physikalische Adresse

ORG $4000 assembliert das Programm nach Adresse $4000. Dort ist es auch lauffähig.

ORG $4000,$6000 assembliert das Programm nach $6000. Lauffähig ist es aber ab Adresse $4000 !

- EQU, EPZ oder "="

Zuweisung einer Konstanten oder eines Rechenergebnisses an ein Symbol.

```
Rtclock:    EQU $14
Rtclock:    EPZ $14 (Equate Page Zero, gleich EQU (vom ATMAS übernommen))!
```

rtclock=$14 wird zu

```
Rtclock:    EQU $14
```

- DFB, DFW

Definieren von Konstanten/Tabellen im Objektcode.

```
Bittab:     DFB 1,2,4,8,16,32,64,128 ; Daten in Bytebreite (8-Bit)
AdressTab:  DFW $b800,$bc00,$9430... ; Daten in Wortbreite (16-Bit)
```

- ASC <STZ>Text<STZ>

<STZ> = String-Trennzeichen (" / \ % $)
" = Text im ASCII-Code einfügen
/ = Inverser ASCII-Code
\ = ASCII-Code, aber (nur) beim letzten Zeichen Bit 7 gesetzt (Ende-kennung)
% = Interner Bildschirmcode
$ = Interner inverser Bildschirmcode

ASC %Dies ist ein %-Zeichen !%

Fügt den Text "Dies ist ein %-Zeichen !" im internen Bildschirmcode in den Objektcode ein. Das 2. String-Trennzeichen wird von hinten nach vorne gesucht, damit das Zeichen selber im Parameter vorkommen kann (der ATMAS macht hier z.B. schwierigkeiten).

- OPT (N,F,R,I)

Option beim übersetzen des Quelltextes.

N = Names
F = Fast
R = Resident
I = Include

Es wird nur das 1. Zeichen erkannt. "OPT Nichts" würde als "OPT Names" interpretiert werden.

Option Names:

Ausgabe einer Symboltabelle mit den Statuswerten (Unused/Macro).

Option Fast:

Assemblieren bei ausgeschaltetem Bildschirm !

Option Resident:

Dieser Assembler ermöglicht das Benutzen einer vordefinierten, einzuladenden Symboltabelle, der sogenannten Resident-List. Diese wird beim BOOTen des Assemblers mit eingeladen, wenn der Schalter "Resident-List" im Preferences-Menü (siehe "Die Menüleiste") auf "On" steht! Sollte also diese Option aktiviert sein, so wird ein Symbol, welches bei seinem Aufruf noch nicht definiert wurde, in dieser 2. Symboltabelle gesucht! Es ist zu beachten, daß der Speicherplatz für die Resident-List von der normalen Symboltabelle abgezogen wird!

Option Include:

Ein Symbol, welches in der Resident-List gefunden wurde, wird in die normale Symboltabelle eingetragen. Das ist notwendig, wenn man z.B. alle benutzten Symbole auf dem Drucker ausgeben möchte (vom Monitor aus möglich), da nur die normale Symboltabelle ausgegeben wird. Diese Option sollte immer mit angegeben werden. Diese Option darf nicht vor der RESIDENT-Option stehen.

Sie können die Optionen durch ein Komma getrennt in einer OPT-Anweisung eingeben ("OPT Fast,R,I") ooder auch einzeln schreiben!

- HLT

Der Befehl HLT bewirkt einen Abbruch des Assemblierungsvorganges an dieser Stelle in der jeweiligen Seite. Der Assembliervorgang wird in der nächsten Seite fortgesetzt (wenn möglich). Diese Anweisung ist sehr nützlich, wenn man "mal eben" schnell was ausprobieren will und nicht den ganzen Text löschen will (oder auch für andere Zwecke).

Beispiel (Seite 1 ist angewählt):

```
 ; Testprogramm
           ORG $600
           OPT N,R,I
 ;
Start:     LDA Random
           STA Wsync
           STA Colpf2
           JMP Start
           HLT  ; Assemblierung in dieser Seite abbrechen !
.
. Hier steht der alte/restliche Quelltext !
.
```

Jetzt kann das Testprogramm ausprobiert werden. Wenn es nicht mehr benötigt wird, kann es mit der Funktion "Block löschen" gelöscht werden. Dann noch alle HLT-Befehle entfernen!

Während der Assemblierung kann mit der BREAK-Taste der übersetzungsvorgang abgebrochen werden.

Funktionen in der Menüleiste:

Die Menüleiste wird vom Editor aus mit der Tastenkombination CONTROL/ESCape aktiviert. In der obersten Bildschirmzeile erscheint dann anstelle der Titelzeile des Assemblers die Menüleiste. Diese Menüleiste enthält 16 Menüpunkte, aus denen mit den Cursortasten oder den Pfeiltasten "größer als" bzw. "kleiner als" einer ausgewählt werden kann. Der gerade ausgewählte Menüpunkt wird invertiert dargestellt. Um die gewählte Funktion zu aktivieren, ist die <RETURN> - Taste zu drücken.

Anmerkung zur File-Auswahl in der Menüleiste:

Sollte irgendeine Diskettenoperation angewählt werden (z.B. "ASCII lesen" oder "Text laden"), so erscheint eine Datei-Auswahlbox. In dieser Box wird ein Teil des Inhaltsverzeichnises der gerade eingelegten Diskette, die Laufwerknummer sowie der freie Speicherplatz auf der Diskette angezeigt.

Zum einladen muß ein File mit den Cursortasten ausgewählt werden und <RETURN> gedrückt werden. Der Filename erscheint jetzt in der unteren Zeile der Box. Zum laden drücken Sie nochmals <RETURN> oder wählen Sie ein anderes File. Die Dateiauswahl kann mit der ESCape-Taste abgebrochen werden.

Zum abspeichern einer Datei wählen Sie den Namen aus dem Inhaltsverzeichnis wie beim laden (zum überschreiben einer bereits existierenden Datei) oder geben Sie einen neuen Namen ein. Dann drücken Sie (nochmals) <RETURN>. Sollte eine abzuspeichernde Datei bereits auf der Diskette vorhanden sein, so werden Sie gefragt, ob Sie die Datei überschreiben wollen. Drücken Sie auf "J" für "Ja" oder auf eine andere Taste, um den Vorgang abzubrechen.

Die Laufwerksnummer können Sie mit "SHIFT" & Zahlentaste wählen, also z.B. "SHIFT/8" für die RAM-Disk. Es erscheinen nicht immer alle Filenamen in der Dateiauswahlbox. Quelltexte im internen Format haben den Extender ".ASM", z.B. "D8:FASTLOAD.ASM", Quelltexte im ASCII-Format haben den Extender ".SRC", z.B. "D1:IOLIB.SRC" ! In den jeweiligen Boxen erscheinen daher auch normalerweise nur Dateien mit der jeweils passenden Extension. Diese Endung (z.B. ".SRC") wird auch in der Dateiauswahlbox angezeigt. Mit der TABulator-Taste kann aber zwischen der normalen Endung und dem kompletten Inhaltsverzeichnis gewechselt werden.

Nun aber zu den Funktionen in der Menü-Zeile:

1. Inhaltsverzeichnis
2. Quelltext laden
3. Quelltext speichern
4. ASCII-Text laden
5. ASCII-Text speichern
6. Text drucken
7. Suche nach ...
8. Sprung zum Disk Operating System (DOS 2.5)
9. Assembler-Toolbox starten
10. Preferences / Voreinstellungen
11. Block merken ("abfotografieren")
12. Block einfügen ("ankleben")
13. Block ausschneiden (löschen und zwischenspeichern)
14. Block löschen ("wegradieren")
15. Quelltext reparieren / retten (Old-Funktion)
16. Quelltext löschen

1. Inhaltsverzeichnis:

- ohne Worte -

2. Quelltext laden:

Es erscheint die Datei-Auswahlbox. Wählen Sie die gewünschte Datei oder brechen Sie die Funktion mit ESCape ab. Sollte die einzuladende Datei kein Quellcode im internen Format sein, wird nichts geladen. Es erscheint auch keine Fehlermeldung, und ein evtl. im Speicher stehender Text bleibt erhalten!

Achtung! Es erscheint auch keine Warnung, falls Sie diesen Menüpunkt versehentlich anwählen und ein im Speicher stehender Text noch nicht gesichert wurde!

3. Quelltext speichern:

Es erscheint eine Datei-Auswahlbox. Wählen Sie eine Datei oder geben Sie einen Namen über die Tastatur ein. Drücken Sie <RETURN>, der Speichervorgang beginnt.

4. ASCII-Text laden:

Dateiauswahl wie bei Punkt 2 (Quelltext laden). Der ASCII-Text wird an der Qursorposition in den evtl. vorhandenen Quelltext eingefügt (entspricht dem INSERT-Modus). Das Laden wird nicht abgebrochen, falls der Speicher nicht ausreichen sollte. Dann geht der Rest des Files allerdings verloren (Es wird immer bis zum EOF (End Of File) gelesen)! Der Programmtext wird während des ladens in das Token-Format (siehe Editor) umgewandelt. Um das Einladen eines inkompatiblen Textes ("andere Assembler - andere Schreibweisen") nicht zu unterbrechen, werden fehlerhafte Programmzeilen (welche nicht ohne Änderungen (Fehler ?) in das Token-Format überführt werden konnten) als Kommentarzeilen behandelt. Beachten Sie bitte die Schreibweise für die High- und LowByte- Auswahl (Beim Macro-Assembler "<" für das Low-Byte und ">" für das HighByte). Beim ATMAS z.B. wird das High-Byte mit einem ":H" (oder ":HI") gekennzeichnet, also z.B. "TABELLE:H". Der Macro-Assembler macht daraus aber "TABELLE ; H", da er den Doppelpunkt als Trennsymbol erkennt!

5. ASCII-Text speichern:

Dateiauswahl wie bei Punkt 3 (Quelltext speichern). Hier gibt es allerdings zwei verschiedene Möglichkeiten: Wenn mit der Blockmarkierungsfunktion ein Textblock markiert wurde, so wird nur der so markierte Bereich abgespeichert. Sonst wird der Text ab der Cursorposition (Zeile, in der der Cursor steht) bis zum Ende der aktuellen Seite gespeichert.

6. Text drucken:

Textmarkierung siehe Punkt 5 (ASCII-Text speichern). Der Text wird auf dem Drucker oder ein anderes Gerät geschrieben (wählbar). Der wohl wichtigste Unterschied ist, das bei diesem Menüpunkt alle Leerzeichen mit ausgegeben werden während bei Punkt 5 der Text im "kurzen" ASCII-Format ausgegeben wird. Voreingestellt als Ausgabegerät ist der Drucker bzw. das im Preferences-Menü eingestellte Gerät. Dieses kann auch ein beliebiger Schnittstellentreiber sein (z.B. "R:600,8N1" für eine RS-232-Schnittstelle) oder auch eine Zieldatei auf Diskette. Die Voreinstellung kann natürlich auch geändert werden.

Beispiel für das Ausgabeformat:

Ausgabe auf Diskette nach Punkt 5 ("_" = Leerzeichen):

```
_Symbol:LDA_(Pointer),Y ; mit Quelle laden
_INY  ; nächster Wert
```

Ausgabe nach Punkt 6 auf ein beliebiges Gerät (formatierte Ausgabe):

```
_Symbol:________LDA (Pointer),Y ; mit Quelle laden
________________INY  ; nächster Wert
```

7. Suchen nach ... :

Der Suchbegriff kann eingegeben bzw. geändert werden. Mit drücken von <RETURN> wird ab der Cursorposition gesucht. Bei erfolglosem Suchvorgang steht der Cursor am Textende, ansonsten wird die Zeile, in der eine übereinstimmung stattfindet, invertiert dargestellt. Nun kann der Suchmodus mit ESCape verlassen oder mit RETURN fortgesetzt werden. Sie können jetzt auch den Suchbegriff ändern und weitersuchen, ohne den Suchmodus verlassen und wieder aufrufen zu müssen. Zum schnellen löschen (zwecks Neueingabe) des Suchwortes können Sie CONTROL/DELETE eingeben! Während der Suche kann man mit der BREAK-Taste den Suchvorgang abbrechen.

8. Sprung zum Disk-Operating-System (DOS 2.5):

Das DOS-Menü wird in den Computer geladen und gestartet. Das DUP.SYS-Menü habe ich speziell für diesen Assembler geändert (Rücksprung zum Assembler durch einfachen Tastendruck). Kopieren Sie das mitgelieferte DOS 2.5 nicht auf anderen Disketten (Kein Rücksprung zum Modul bzw. BASIC) !

9. Assembler-Toolbox starten:

Bei diesem Assembler ist es möglich, eigene Programme oder Erweiterungen bequem vom Editor aus aufzurufen (z.B ein Packer o.Ä.) ! Ein Beispiel- Sourcefile finden Sie im File "TOOLBOX.ASM" auf der Assembler-Diskette. Dieses Programm muß so assembliert werden, daß es ab Adresse $f400 lauffähig ist. Am besten, Sie benutzen das Beispielprogramm und ergänzen das Menü und die Tastaturabfrage. Das so erzeugte Programm muß mit dem Write-Befehl des Monitors (nicht mit der Save-Funktion) unter dem Namen "TOOLBOX.UTL" abgespeichert werden. Dieses File wird beim Booten des Assemblers mit in den Speicher geladen und steht dort solange, bis es durch eine anwachsende Symboltabelle überschrieben wird oder bei Aufruf der Toolbox mit drücken von <RETURN> die <SHIFT> - Taste gedrückt wird. Dann wird versucht, die Toolbox nachzuladen !

In der Toolbox des Assemblers V4.3 (Toolbox-Version 1.1) habe ich zwei Tools programmiert. Es können damit alle Bemerkungen aus dem Quelltext entfernt werden bzw. die Parameter optimiert werden.

Beispiel zur Parameter-Optimierung:

```
Tabelle:        DFB %001001,$09,$0f,0123,Test+(%01!%010)
Start:          LDA #$0f
```

wird zu

```
Tabelle:        DFB %1001,9,$f,123,Test+(1!%10)
Start:          LDA #$f
```

10. Preferences / Voreinstellungen:

Hier können Sie folgende Einstellungen vornehmen:

1 - Printer-Device (Default-Gerät bei Anwählen des Drucker-Symbols).
2 - Farben einstellen (mit "<", ">", "+" und "-").
3 - Den Zeichensatz wählen, der nach dem Booten eingeschaltet sein soll.
4 - Den Directory-sort-Index wählen.
5 - Das INVERSE-Menü-Flag setzen.
6 - Den IO-Sound An- oder Abschalten.
7 - Die Resident-List An- oder Abschalten.

Die geänderten Einstellungen können mit "Save Prefs" abgespeichert, mit "Quit Prefs" nur benutzt oder mit "Load Prefs" die Preferences-Datei geladen werden.

1 - Printer-Device:

Wenn Sie keinen Drucker haben, den Sie über den Gerätetreiber "P:" ansprechen können, aber z.B. ein "RS-232"-Treiber haben, so können Sie diesen vom Monitor aus einladen und initialisieren. Tragen Sie dann hier die Gerätetreiberbezeichnung ein (z.B. "R:300,8N1").

2 - Farben einstellen:

Stellen Sie hier die Farben ein, welche der Assembler haben soll.

3 - Zeichensatz wählen:

Die Symboltabelle überschreibt schon bei sehr kurzen Programmen den originalen Zeichensatz (im RAM). Damit es bei der Assemblierung oder bei sonstigen Zugriffen auf die Symboltabelle nicht zu einem unschönen Flackern kommt, weil die Daten aus der Symboltabelle als Zeichensatzdaten interpretiert werden, kann hier der Zeichensatz mit den deutschen Umlauten/Sonderzeichen eingeschaltet werden. Da einige aber auch eine Happy-Erweiterung besitzen (Floppyspeeder), deren Parallel-Treiber im Sonderzeichensatz-Bereich im ROM liegt, kann man auch den Originalzeichensatz (Standart-Font) wählen.

4 - Directory-Sort-Index:

Hier kann man einstellen, ob mit der Sortierung des Directorys beim 1. Zeichen (0) oder beim 2. Zeichen (1) begonnen werden soll. Das 1. Zeichen ist z.B. der Stern "*" für schreibgeschützte Dateien. Sollte also "0" gewählt sein, so werden die schreibgeschützten Dateien extra unter den "normalen" Dateien angezeigt. Ein Bindestrich zeigt an, daß das Directory nicht sortiert werden soll.

5 - Inverse-Menü:

Abhängig von den gewählten Farben ist es nötig, dieses Flag zu setzen. Dies hat nur einen optischen Sinn (ausprobieren). Die Auswirkung dieses Flags werden allerdings erst beim nächsten Aufruf der Menüleiste sichtbar.

6 - IO-Sound An-/Abschalten:

Mit diesem Flag kann das Ladegeräusch aus- oder eingeschaltet werden. Einige Speeder berücksichtigen dieses Flag allerdings nicht.

7 - Resident-List An-/Abschalten:

Wenn die Resident-List beim Aufruf von Preferences abgeschaltet war und Sie die Resident-List hier einschalten, wird sie beim verlassen mit "Quit Prefs" nachgeladen! Zur weiteren Information lesen Sie bitte die Beschreibung des Assemblers !

Bei den Punkten 3 - 7 wird die Einstellung mit <SPACE> geändert. In der letzten Zeile wird mit <SPACE> zwischen "Load Prefs", "Save Prefs" und "Quit Prefs" umgeschaltet. Ein Punkt kann mit <RETURN> oder ESCape übersprungen werden (Bei Load, Save oder Quit Prefs führt <RETURN> zur Ausführung der Funktion (z.B. Save Prefs) und dann zu einer Rückkehr in den Editor, während man mit ESCape wieder bei Punkt 1 landet).

Wird ein Block z.B. ausgeschnitten, So wird er, da kein Platz mehr im Rechnerspeicher ist, auf Diskette bzw. RAMDisk geschrieben. Bei Atari XE Computern oder ATARI 800 XL mit 128 kByte RAM (64 kB RAMDisk) kann die RAMDisk nicht mehr vom DOS aus angesprochen werden, da dort Programmteile und Quelltexte "liegen", während bei XL's mit z.B. 256 kB RAMDisk auf die RAMDisk (64 kB) vom DOS aus zugegriffen werden kann. Sollte der Speicher auf der Diskette (oder RAMDisk) nicht mehr ausreichen, wird eine Fehlermeldung (Fehlercode für "Disk Full") ausgegeben ! Der Name der Datei ist "D8:CLIP.BLK" oder "D1:CLIP.BLK" ! Sollte kein Block markiert sein, so wird (außer bei "Block einfügen") eine entsprechende Fehlermeldung ausgegeben.

<u>11. Block merken:</u>

Der markierte Block wird in den Block-Puffer abgespeichert, ohne den Quelltext zu verändern.

<u>12. Block einfügen:</u>

Ein vorher mit "Block merken" oder "Block ausschneiden" gespeicherter Block wird ab der Cursorposition in den Quelltext eingefügt. Sollte die Block-Datei länger sein als Speicherplatz zur Verfügung steht, so erscheint eine Fehlermeldung.

<u>13. Block ausschneiden:</u>

Der markierte Block wird in den Block-Puffer abgespeichert und danach aus dem Quelltext gelöscht.

<u>14. Block löschen:</u>

Der markierte Block wird aus dem Quelltext unwiederbringlich gelöscht. Der Block-Puffer wird dabei nicht verändert (ein vorher abgespeicherter Block bleibt also erhalten).

<u>15. Quelltext reparieren (gelöschten Quelltext zurückholen):</u>

Mit diesem Menüpunkt kann ein versehentlich gelöschter Quelltext gerettet werden. Der Quelltext wird als doppelt verkettete Liste gespeichert. Jede Zeile beginnt daher mit einem $ff-Byte (dem Zeilen-Header), gefolgt von einem Link-Byte zur vorherigen Zeile und einem Link-Byte zur nachfolgenden Zeile. Danach folgt der (codierte) Programmtext. Sollte zum Beispiel ein Linkbyte nicht mehr auf den Zeilen-Header zeigen, kommt es beim Editor zum fehlerhaften Bildaufbau, es erscheint Schrott auf dem Bildschirm, der Assembler meldet einen logischen Fehler. Sollte dieser Fehler auch nach dem Reparieren des Quelltextes noch vorhanden sein, so ist die zeileninterne Codierung (Tokencodes, Längenangaben von Parameterblöcken usw.) durcheinander geraten. Dann muß die jeweilige Zeile von Hand geändert werden. Dieses Fehlverhalten kann auftreten, wenn z.B. ein eigenes Programm abgestürzt sein sollte oder wenn während eines Speichertransfers (Eine eingegebene Zeile wird sofort in den Quelltext eingefügt) die RESET-Taste gedrückt wird und so der Speichertransfer unterbrochen wurde. Dann können einige Linkbytes auf falsche Speicherstellen zeigen.
Sollte ein eigenes Programm mal abstürzen und Teile des Assemblers zerstören (das ist möglich!), so sollte sofort ein Kaltstart durchgeführt werden. Dazu NICHT den Computer ausschalten, sondern die Funktionstasten OPTION, SELECT und START gedrückt halten (!) und RESET drücken. Der Bildschirm muß jetzt für zwei bis drei Sekunden blinken, danach wird ein Kaltstart durchgeführt! Der Quelltext wird von dem daraufhin stattfindenden Speichertest ja nicht zerstört! Jetzt können Sie den Assembler laden und den Quelltext retten, wenn er nicht durch einen schwerwiegenden Absturz zerstört wurde.

<u>16. Quelltext löschen:</u>

Der Quelltext wird ohne Sicherheitsabfrage gelöscht.

Anmerkung zu Punkt 15 und 16:

Diese Funktionen wirken jeweils nur auf die angewählte Seite (Quelltextebene) und müssen gegebenenfalls auf mehrere Seiten angewendet werden!

Der Monitor (Version 3.3):

In den Monitor gelangt man vom Editor aus mit "CONTROL/P" !

Speichern und Laden / Inhaltsverzeichnis:

Der Monitor hat folgende Befehle zum speichern und laden von Daten:

Load: L <Filename>
Save: S Von,Bis,<Filename>(,Ladeadresse)
Read: R <Filename>,Nach(,Länge)
Write: W Von,Bis,<Filename>

Zum Anzeigen des Disketteninhaltes gibt es den Befehl
I (DriveNum) oder I <Filename>.

Mit "L <Filename>" wird ein Binärfile in den Speicher geladen. Dabei werden die jeweils belegten Speicherbereiche angezeigt mit "FROM xxxx TO yyyy", "INIT AT xxxx" oder "AUTORUN xxxx". In den letzten beiden Fällen (Init/Autorun) wird auf einen Tastendruck gewartet. Mit <RETURN> wird zur INIT bzw. AUTORUN-Adresse gesprungen. Nach ordentlicher Rückkehr, oder wenn eine andere Taste gedrückt wurde, wird mit dem Laden fortgefahren.

Mit "S Von,Bis,<Filename>(,Ladeadresse)" wird der Speicherbereich von "Von" bis "Bis" nach <Filename> als Binärfile abgespeichert.

Beispiel:
"S $4000 $43ff D:TEST.OBJ"

"S $4000 $43ff D:TEST.OBJ $8000"

Im 2. Beispiel wird der Bereich von $4000 bis $43ff so abgespeichert, daß er nach $8000 bis $83ff geladen wird.

Mit "R D:TEST.DAT $4000" werden die Daten aus dem File TEST.DAT gelesen und ab Adresse $4000 abgelegt.

"R D:TEST.DAT $4000 $100" ließt nur $100 Bytes aus dem File TEST.DAT nach $4000 bis $40ff !

"I", "I 1" oder "I D1:*.DAT" Zeigt das Inhaltsverzeichnis der Diskette an.

Starten von Maschienenprogrammen:

G Adr(,A(,X(,Y)))

"G START 120,,TEST" startet ein Maschienencodeprogramm an der Adresse "START" (Symbol) und übergibt im ACCU den Wert 120 und im Y-Register den Wert des Labels "TEST". X enthält einen undefinierten Wert! Das Programm kann auf drei Arten beendet werden: Mit einem RTS-Befehl, einem BRK-Befehl oder mit der Tastenkombination "SHIFT/BREAK".

Wird das Programm mit einem "RTS" beendet, so wird eine entsprechende Meldung ausgegeben. Bei einem Abbruch mit BRK oder SHIFT/BREAK werden die Register sowie die Statuswerte ausgegeben. Mit "C" oder "C Adr" kann man im Programm fortfahren (Continue-Funktion).

M Von,Bis,Nach
F Von,Bis,Mit

"M $3000,$4fff,$4000" verschiebt den Speicherbereich von $3000 bis $4fff nach $4000 bis $5fff.

"F $4000 $7fff $80" Füllt den Speicherbereich von $4000 bis $7fff mit dem Wert $80.

Disassembler:

D (Start(,Ende(,<Filename>)))

"D (Adr)" gibt 15 Zeilen aus, um dann auf einen Tastendruck zu warten. Wird "X" oder "ESCape" gedrückt, wird das Disassemblieren abgebrochen, sonst werden weitere 15 Zeilen ausgegeben.

"D $a800 $abff P:" gibt ein Listing auf dem Drucker aus. Auf Tastendruck hält das Listen an, nochmaliger Tastendruck außer "X" und "ESCape" läßt es weiterlaufen.

Speicherauszug/Speicherinhalt ändern:

mit ".(Adr)" werden 13 Zeilen des Speicherbereiches als HEX-Dump und als ASCII ausgegeben. Wird <RETURN> mit SHIFT gedrückt, wird anstatt ASCII Bildschirmcode gelistet.

Mit ".$8000 1 2 4 8" werden die Werte (nur 8 Bit) 1,2,4,8 ab Adresse $8000 abgelegt.

Berechnung und Umrechnung:

"?START,ENDE-START" zeigt den Wert des Symbols "START" sowie das Ergebnis der Rechenoperation "ENDE minus START" an.

Zurück zum Editor:

Mit "X" (für Exit) kommt man wieder in den Editor!

Ausgabe der Symboltabelle:

"V" zeigt die komplette Symboltabelle an.

"V *" zeigt die Symboltabelle "halbwegs" sortiert an (Erst alle Symbole mit A, dann mit B,....) !

"V abc*" Zeigt die Symbole, welche mit "ABC" anfangen, sortiert an.

"V #P:" oder "V #P: *" gibt die Symboltabelle auf dem Drucker aus !

Sie können mit "V #D:TEST.LST" die Symboltabelle in ein File schreiben und dieses auch mit der "ASCII Text lesen"-Funktion vom Editor einladen (beachten Sie, das im Directory nur *.SRC-Files angezeigt werden und wählen Sie mit TAB das "Extended Directory")!

- Wenn ein Befehl einen numerischen Parameter benötigt, kann selbstverständlich auch ein Symbol angegeben werden.

- Wenn die Menü-Leiste vom Editor aktiviert wird, wird die Symboltabelle gelöscht (nicht die Resident-List)!

Fehlermeldungen des Editors:

Adressierungsart

- Die angegebene Adressierungsart für diesen Befehl ist unbekannt. Sie haben entwender die falsche Adressierungsart eingegeben (z.B. "LDA (10,Y)") oder den Parameter vergessen (entspricht Adressierungsart "Accu")!

Labelende falsch

- Sie haben den "Klammeraffen" ("@") mitten in einem Symbol verwendet. Dieses Zeichen darf nur am Ende eines Symbols stehen und kennzeichnet es als lokales Label. Wenn Sie ein Label (Sprungziel) in einem Symbol verwenden, ohne dieses als "Local" zu kennzeichnen, erhalten Sie beim 2. Aufruf des Macros die Fehlermeldung "Doppelt benutzt"! Das "@" wird durch eine vierstellige Nummer ausgetauscht, welche bei jedem Macro-Aufruf um den Wert "1" erhöht wird !

Label fehlerhaft

- Sie haben in einem Sprungziel fehlerhafte Zeichen (z.B. Rechenoperatoren usw.) eingegeben. Erlaubt sind nur a-z, A-Z, 0-9, "." und "_" (und "@"). Sie können also auch Namen wie "Test.Prg" oder "Stick_1" verwenden.

Symbol fehlt

- Sie haben einen Befehl eingegeben (z.B. "EQU 10"), ohne diesem ein benötigtes Symbol voranzustellen. Diese Meldung tritt auch auf, wenn Sie nur einen Doppelpunkt eingeben. Ein Doppelpunkt kennzeichnet das Ende eines Symboles.

Parameter fehlt

- Sie haben einen Befehl eingegeben (z.B. "rtclock EQU"), ohne den benötigten Parameter (Ausdruck) mit anzugeben.

Macro: Name fehlt

Sie haben die Macro-Kennzeichnung (".") oder eine Macro-Definition ("Name: MAC") ohne Macro-Namen eingegeben. Sie müssen z.B. ".Name" eingeben (nicht etwa ". Name") !

Macro: fehlerhaft

- Sie haben ein Macro falsch aufgerufen. Wenn Sie z.B "A *1" eingeben, nimmt der Editor das A als Symbol und das "*1" als Macro-Name, da er beides keinem Befehl zuordnen kann. Ein Macro-Name muß mit einem der unter "Label fehlerhaft" beschriebenen Zeichen beginnen.

Nur ohne Symbol

- Der eingegebenen Assembler-Directiven (z.B. "ORG") darf kein Symbol vorangestellt werden. Einem Label (z.B. "Adress:") wird beim Assemblieren der Wert des Adresszählers zugewiesen. Dieser Adresszähler wird z.B. bei "ORG" verändert. Der Assembler kann aber nicht wissen, ob er dem Symbol den Wert des Adresszählers vor oder nach dessen Änderung zuweisen soll. Auch bei "HLT" darf kein Symbol mit angegeben werden.

Was'n das ???

- Der Editor kann mit der Eingabe nichts anfangen. Sie haben absoluten Schrott eingegeben (oder z.B. zwei Doppelpunkte hinter einem Symbol).

Fehlermeldungen des Assemblers:

Folgende Meldungen erscheinen erst beim Assemblieren:

Doppelt benutzt

- Sie wollen ein Symbol definieren, welches schon existiert. Diese Meldung kann auch auftreten, wenn ein Symbol aus der Resident-List (Bei eingeschalteter Option "Resident" und "Include") benutzen und erst später definieren. Wenn Sie einem Symbol, was in der Resident-List existiert, einen neuen Wert zuweisen wollen, müssen Sie das vor der ersten Benutzung erledigen! Das Symbol wird nicht in der Resident-List verändert, sondern nur mit dem neuen Wert in die Symboltabelle eingetragen! Symbole werden immer zuerst in der Symboltabelle gesucht!

Symboltab. voll

- Die Symboltabelle ist mit 10 K-Bytes sehr großzügig bemessen. Entweder Sie haben irre lange Namen benutzt oder aber zu viele Macro-Aufrufe mit Localen Symbolen (Aus "Locales_Label_@:" wird "Locales_Label_0001","Locales_Label_0002" usw.). Sie müssen in Ihren Macro-Definitionen die Namen der localen Label kürzen! Wenn Sie gar keine Macros benutzt haben (oder keine localen Symbole), haben Sie wirklich irre lange Symbole verwendet! Kürzen ist angesagt!

Symbol zu lang

- Ich bin dafür, das die Namen der Symbole so gewählt werden, das ihr Sinn in dem jeweiligen Programmteil auch erkannt wird (z.B. "CopyLoop1:", schlimmstenfalls noch "Cpl1:" oder so). Ein Symbol darf maximal 59 (!) Zeichen lang sein. Sie haben wahrscheinlich "Dieses_hier_ist_meine_Daten_Kopierschleife_nummer_eins:" verwendet! Lange Symbole erhöhen die übersetzungszeit und den Speicherbedarf!

Symbol unbekannt

- In einem Ausdruck verwenden Sie ein Symbol, welche nirgends definiert wurde. überprüfen Sie die beanstandete Zeile auch auf Tippfehler. Vielleicht haben Sie das Symbol auch bei seiner Definition falsch geschrieben!

Ausdruck falsch

- Der angegebene Ausdruck enthält einen Fehler (Falscher Operator), ist unvollständig (nicht alle oder zu viele Klammern geschlossen) oder die "ASC"-Anweisung enthält falsche String-Trennzeichen oder einen LeerString (ASC "" oder "Test%)!

Logischer Fehler !!!

- Der logisch Aufbau des Quelltextes (in der bemängelten Zeile) wurde irgendwie zerstört. Siehe auch Beschreibung der Menüleiste/Punkt 15 (Quelltext reparieren). Diese Meldung kann auch bei Timing-Problemen der RAMDisk auftreten. Dann besteht allerding höchste (Absturz-) Gefahr !!! Versuchen Sie die Zeile neu einzugeben oder zu editieren! Dieser Fehler darf normalerweise nicht vorkommen!

Bisher kein ORG

- Sie müssen mit ORG die Adressenlage des zu erzeugenden Objectcode bestimmen, bevor Sie Daten/Befehle in den Speicher schreiben wollen !!!

- Im zweiten Durchgang (Pass 2) wurde ein anderer OpCode erzeugt als beim ersten. Dieser Fehler tritt auf, wenn Sie eine ZeroPage-Adressierung benutzen, das entsprechende Symbol aber erst später definieren. Ist zum Beispiel beim übersetzen der Zeile "LDA Adr,X" das Symbol "Adr" noch undefiniert, wird automatisch die Adressierungsart "Absolut-X" (Adresse in Wortbreite) genommen! Wird das Symbol "Adr" später mit z.B. "Adr: EQU $e4" definiert (Zero-Page!), so erkennt der Assembler im zweiten Durchlauf eine "ZeroPage-X"-Adressierung (Adresse in "Bytebreite" (8 Bit)) und meldet diesen Fehler. Da die ZeroPage-Adressierung kürzer ist, stimmen ab diesem die restlichen Sprungziele nicht mehr! Sie müssen das Symbol vor der ersten Benutzung definieren. Zu einem übersichtlichen und guten Programmierstil gehört sowieso, das möglichst alle Symbole am Anfang eines Programmes definiert werden!

Abbruch mit BRK

- Sie haben die übersetzung mit der BRAK-Taste abgebrochen!

Ziel zu weit

- Relative Sprünge (BNE, BPL usw.) können nur um +129/-125 Bytes springen! Sie müssen JMP verwenden.

Zahl zu gross

- Eine im Ausdruck angegebene Konstante läßt sich nicht als Wort (16 Bit) darstellen. überprüfen Sie den Ausdruck (Sie können zwar "$0000e474" angeben, aber nicht "00000034") !

Adressueberlauf!

- Der Adresszähler hat einen überlauf. Sie haben eine fehlerhafte ORG-Adresse angegeben oder der Objectcode ist zu lang.

Zu komplex

- Der Ausdruck ist zu verschachtelt (mehr als 7 geöffnete Klammern) oder enthält zu viele Parameter. Sie können maximal 32 numerische Parameter in einer DFB- oder DFW-Anweisung haben. Wenn ein Ausdruck berechnet wird, wird er zuerst in 16-Bit-Werte zerlegt und danach erst ausgerechnet!

Division durch 0

- Durch Null kann nicht dividiert werden!

Macro-Fehler

Diese Fehlermeldung tritt auf, wenn

- die Anzahl der beim Macro-Aufruf übergebenen Parameter nicht mit der in der Macro-Definition festgelegten Anzahl übereinstimmt.
- die Macro-Parameterfunktion außerhalb einer Macro-Definition verwendet wird.
- in einer Macro-Definition auf einen Macro-Parameter zugegriffen wird, der garnicht übergeben wurde.
- ein Macro-Name in einem Ausdruck verwendet wird.
- eine Macro-Definition nicht in der gleichen Textseite geschlossen wird, in der sie eingeleitet wurde.

- eine neue Macro-Definition eingeleitet wird, ohne eine vorherige zu schließen.

- versucht wird eine Macro-Definition zu schließen, obwohl keine geöffnet wurde.

- ein Macro-Aufruf in einer Macro-Definition erfolgt.

- Befehle wie z.B. "HLT" oder "ORG" in Macro-Definitionen benutzt werden.

- wenn Sie mehr als 9999 Macro-Aufrufe in Ihrem Pogramm verwenden.

<u>Erläuterungen zum Umgang mit Macros:</u>

Eine Macro-Definition (-Einleitung) beginnt mit dem Namen und dem Befehl "MAC", gefolgt von der Anzahl der zu übergebenden (numerischen) Parameter. Sollen keine Parameter übergeben werden, können Sie auch "Name: MAC" statt "Name: MAC 0" eingeben. Beendet wird eine Macro-Definition mit dem Befehl "END". Parameter werden immer in 16-Bit-Breite übergeben. Die Abfrage der Parameter erfolgt mit dem Fragezeichen und der Nummer des Parameters. Beispiel:

```
 ;
 ; Poke-Macro (wie der BASIC-Befehl POKE ADRESSE,WERT)
 ;
Poke:     MAC 2 ; 2 Parameter übergeben
          LDA #?2 ; Accu mit 2. Parameter laden (Wert)
          STA ?1 ; und zum 1. Wert schreiben (Adresse)
          END  ; Ende der Macro-Definition
 ;
          ORG $3000
 ;
Start:    .Poke $bf00+10*10,129
          RTS
 ;
```

Wenn Sie dieses Programm assembliert haben, können Sie sich das Ergebnis vom Monitor Disassemblieren lassen. Gehen Sie mit CONTROL/P in den Monitor und geben Sie "D START" ein. Es wird "LDA #$81" (129) und "STA $BF64" ($bf00+100) ausgegeben (u.A.) ! Wenn Sie versuchen auf den 3. Parameter (z.B. mit "LDA #?3") zuzugreifen, erhalten Sie die Fehlermeldung "Macro-Fehler", da nur Zwei zu übergeben waren! Sie können mit den Parametern auch rechnen! Beispiel:

```
 ;
 ; Wie mit Macro-Parametern gerechnet wird
 ; (Siehe IOLIB.SRC - .Close <Kanal>
 ; Altes (Original-) Macro:
 ;
Close:     MAC 1
           LDA #?1 ; Kanal-Nummer
           ASL  ; Multiplikation mit 16 (Wert*2*2*2*2)
           ASL
           ASL
           ASL
           TAX  ; Index für Kanal
           LDA #Cclse
           STA Iccom,X
           JSR Ciov
           END
 ; Kürzere Version:
Close:     MAC 1
           LDX #?1*16
           LDA #Cclse
           STA Iccom,X
           JSR Ciov
           END
```

Nachtrag:

In der Toolbox habe ich eine neue Funktion programmiert. Mit Punkt 3 (Umschalten) kann bei 256 kB-Erweiterungen zwischen zwei verschiedenen Quelltexten umgeschaltet werden.

Wenn Sie die "Resident-List" verändern wollen, vergessen Sie nicht die Zeile "Ende: DFB 0" !!! Speichern Sie immer vom Monitor mit dem Write-Kommando "W START ENDE D:RESLIST.DAT"! Erzetzen Sie ENDE nicht durch die vom Assembler angegebene physikalische Endadresse! "ENDE" bezeichnet das Label !!!

Der Unterstrich kann nur eingegeben werden, wenn die Zeile editiert wurde. "SHIFT/-" scrollt normalerweise um 16 Zeilen. Wenn Sie also aus "Test.Label:" z.B. "Test_Label:" machen wollen, müssen Sie ein anderes Zeichen eingeben und dieses überschreiben.

Der Macro-Assembler läuft nur mit dem geänderten DOS 2.5 einwandfrei (andere Versionen wurden nicht getestet)! Achtung: Beim Konfigurationstest (wenn das Titelbild beim laden erscheint) werden direkt im DOS.SYS Speicherinhalte geändert (Anpassung für "D8:MEM.SAV" bzw. "D8:DUP.SYS") ! Wenn Sie ein anderes DOS benutzen, können Daten auf der Diskette unwiederbringlich zerstört werden da andere DOS-Versionen allergisch auf diese Änderungen reagieren können (So wird beim Booten das DUP.SYS zwar in die RAMDisk kopiert, aber bei nur 64 KB-RAMDsik ein Zugriff durch diese nachträglichen Änderungen auf die RAMDisk unterbunden ! Sonst würde ein versehentliches Speichern auf die RAMDisk den Quelltext oder Assembler zerstören (und mann könnte nicht auf das DUP.SYS zugreifen) ! Verwenden Sie das mitgelieferte DOS nicht auf anderen Disketten !

Das Programm (und die Anleitung) ist Shareware. Es darf nur kostenlos weitergegeben werden (wie Public Domain). Es wäre Fair, wenn regelmäßige Benutzer mir einen kleinen Betrag zukommen lassen würden! Ich bin hier Ausdrücklich dagegen, dieses Programm z.B. gegen "eine Kopiergebühr von nur 10,-- DM" oder so weiterzugeben !!! Ich finde es echt fies, wenn andere an PD-Programmen kassieren und der Programmierer leer ausgeht ! Das ist ja schon schlimmer als Raubkopieren !

Mitgelieferte Utilities:

FirstAid

FirstAid ist in ATARI-BASIC geschrieben. Es rekonstruiert aus einem mit dem PowerPacker V2.1 gepacktem Programm die ungepackte Originaldatei. Sollte es vorkommen, das ein Programm gepackt nicht laufen sollte (dürfte es nicht geben), so kann man hiermit die Originalversion erstellen. (Falls ein Programm über Init gestartet wird und den Entpacker "beschädigt").

SpeedFix

SpeedFix installiert auf XL/XE-Computern die Speedy-High-Speed-SIO-Routine. Das OS-ROM wird ins RAM kopiert und die HS-Routine in den 2. internationalen Zeichensatz geladen. Anschließend wird die Sprungtabelle geändert.

Es funktioniert nicht mit Programmen, welche das RAM unter dem OS-ROM belegen !!! Absturzgefahr!

Es kann vom DOS-Menü geladen werden.

PACKER (XL Packer hat mehr Speicher!)

Mit dem Packer können fast alle Binary-Load-Files in ihrer Länge verkürzt werden. Die zu packenden Programme müssen selbststartend sein. Außerdem können selbstgeschriebene Programme (z.B. Vorspann, Zeichensatz, Hauptprogramm usw.) zu einem einzelnen Programm "zusammengelinkt" werden.

Nach dem laden meldet sich der Packer mit der Kommandozeile. Es gibt die Kommandos Dir, Load, Info, Pack, Rest. (Restore), Save, Quit sowie New.

- Dir

  Inhaltsverzeichnis ausgeben.

- Load

  Binärfile einladen. Das einzuladende Binärfile kann auch mehrteilig sein. Es kann auch eine INIT-Adresse angegeben werden.

  Beispiel:
  Load - Filename(,Init)

- D:FONT.BIN normales einladen

- D:TITEL.COM, einladen und Init-Vektor an 1. Adresse erzeugen

- D:TITEL.COM,A800 einladen und Init-Vektor auf angegebene Adresse erzeugen.

- ,1F00 Init-Vektor erzeugen oder austauschen.

  Sollte ein Init-Vektor schon existieren, wird die Adresse ausgetauscht (nur beim letzten Eintrag)!

- Info

  Es wird die Speicherbelegung aller geladenen Files angezeigt. Von gepackten Teilen wird die Originallänge und die gepackte Länge angezeigt.

Beispiel:
FROM:8000 TO:82A4 LEN:02A5
Init at:8000
FROM:0400 TO:0477 LEN:0078
FROM:1F00 TO:3EA0 LEN:2FA1 / 22EB   Original / gepackte Länge
FROM:00F0 TO:00F6 LEN:0007
Init at:0400
Autorun:1F00

Zum Schluß wird noch die Gesammtlänge angezeigt (normale Filelänge und gepackte Filelänge (mit Entpacker)).

- Pack

Im Speicher stehende ungepackten Programmteile werden gepackt.

- Rest.
- Restore packed Data -
Im Speicher stehende gepackte Programmteile werden "entpackt".

- Save
Abspeichern unter dem angegebenen Namen.
Beispiel:
Save - Filename(,Autorun(,Relocator)):

- D:AUTORUN.SYS,A800
Das (gepackte) Programm wird unter dem Namen "AUTORUN.SYS" gepeichert und ein Autostart-Vektor an die angegebene Adresse (Hier A800) erzeugt. Der Entpacker liegt im Kassetten-Buffer (0400-0477)

- D:SPIEL.COM,1F00,8000
Das Programm wird gespeichert, ein Autorun-Vektor nach 1F00 erzeugt sowie der Entpacker nach 8000 reloziert, liegt also beim späteren Einladen des Programmes nicht im Kassettenbuffer (Entpacker enthält absolute Sprünge und Unterprogrammaufrufe)!

- D:SPIEL.COM,,0100
Das Programm wird gespeichert und der Entpacker nach 0100 reloziert (CPU-Stack). Ein Autorun-Vektor wird nicht erzeugt, muß also schon im File vorkommen!!!

Sollte kein Programmteil gepackt sein, so wird auch kein Entpacker mit in das File geschrieben (normale Link-Funktion).

- Quit
Verlassen des PowerPackers nach Sicherheitsabfrage (ins DOS).

- New
Löschen des Speichers für einen neuen Packvorgang.
Wenn Sie mehrere Programme packen wollen, müssen Sie den Speicher nach dem Speichern wieder löschen, sonst "linken" Sie zwei Programme!

Beispiele:

- Es sollen fertige Programme gepackt werden:

Evtl. Speicher mit "N"ew löschen!
"L" drücken
Load - Filename(,Init)
"D:SPIEL.OBJ"
FROM:8000 TO:BFFF LEN:4000   (Programmdaten)
FROM:02E0 TO:02E1 LEN:0002   (Autorun-Vektor)

Evtl. mit "I" die Speicherbelegung anzeigen lassen.

"S" drücken
Save - Filename(,Autorun(,Relocator))
D:SPIELP.COM
Saving ...
FROM:0400 TO:0477 LEN:0078 (Entpacker wird geschrieben)
FROM:8000 TO:AC4E LEN:2C4F (Gepackte Daten)
FROM:00F0 TO:00F6 LEN:0007 (Informationen für den Entpacker)
New Vector Jump (Init/Autorun) (Starten des Entpackers)
FROM:02E0 TO:02E1 LEN:0002 (Autostart für Programm)

File saved!

Wenn man ein eigenes Programm geschrieben hat, ergibt sich das Problem, daß es aus mehreren Teilen besteht, z.B. Zeichensatz, Hauptprogramm, Titelvorspann, Spritedaten (tschuldigung, ich meine Player-Missile-Daten...) usw. und so fort!
Diese Dateien möchte man (sollte man möchten...) gerne in einer einzigen Programmdatei haben, also z.B. nicht extra den Zeichensatz nachladen.
Dazu ist es allerdings nötig, das alle Dateien im Binary-Load-Format vorliegen ($FFFF - Startadresse - Endadresse - Daten - ...)! Sollten Sie zufällig über den Macro-Assembler XE, V4.3 verfügen, so können Sie den Zeichensatz mit der "R"ead-Funktion einlesen und mit "S"ave abspeichern (z.B. "R D:ZEICHENS.FNT $3000" und "S $3000 $33FF D:ZEICHENS.BIN Zieladresse") !

Angenommen, Sie haben folgende Dateien, welche Sie gerne "linken" wollen:

TITEL.COM	Zeigt ein Titelbild, muß an 1. Adresse gestartet werden
ZEICHENS.BIN	Zeichensatz-Daten
SPRITES.BIN	"Sprite"-Daten (PM-Daten)
DSOUND.BIN	Sound-daten (digitalisierte Sound-effekte)
BALLER.OBJ	Das eigentliche Spielprogramm

1. Titel.Com laden:

"D:TITEL.COM," (Komma nicht vergessen, evtl. Adressangabe)
Es erscheint die Meldung "from-to-Len" sowie "Init installed!"

2. Zeichensatz:
"D:ZEICHENS.BIN"
Es erscheint die Meldung "from-to-len" (ohne Init-Kram!).

3. Sprite-Daten:
"D:SPRITES.BIN"

4. Sound-Daten:
"D:DSOUND.DAT"

5. Hauptprogramm:
"D:BALLER.OBJ"

Wenn es notwendig sein sollte, eventuelle Interrupts in der Titelbildroutine auszuschalten, bevor das Hauptprogramm gestartet wird, müssen Sie noch einen Init-Vektor auf die ausschaltende routine installieren, welche in TITEL.COM stehen sollte !
",3403" (als Beispiel)

Jetzt mit "P" Packen (oder auch nicht).

Jetzt Abspeichern mit "S":
Save - Filename(,Autorun(,Relocator)):
D:SPIEL.COM,"Autostartadresse"(,evtl. neue Entpackeradresse)
z.B."D:BALLER.COM,8000,0100".

Nicht ohne Autostart abspeichern!

überschreiben Sie NIEMALS eine Originaldatei !!!